Manchester Metropolitan University

e-space

Manchester Metropolitan University's Research Repository

Please cite the Published Version

Khalfay, Amy, Crispin, Alan and Crockett, Keeley (2020) Solving the service technician routing and scheduling problem with time windows. In: Intelligent Systems Conference (IntelliSys) 2019, 05 September 2019 - 06 September 2019, London, United Kingdom.

DOI: https://doi.org/10.1007/978-3-030-29516-5_86

Publisher: Springer

Version: Accepted Version

Downloaded from: https://e-space.mmu.ac.uk/622782/

Usage rights: © In Copyright

Additional Information: This version of the article has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: http://dx.doi.org/10.1007/978-3-030-29516-5_86

Enquiries:

If you have questions about this document, contact openresearch@mmu.ac.uk. Please include the URL of the record in e-space. If you believe that your, or a third party's rights have been compromised through this document please see our Take Down policy (available from https://www.mmu.ac.uk/library/using-the-library/policies-and-guidelines)

Solving the service technician routing and scheduling problem with time windows

Dr Amy Khalfay[1], Dr Alan Crispin[2], and Dr Keeley Crockett[2]

[1] IBM Services, Global Business Services amy.khalfay@ibm.com
[2] Manchester Metropolitan University
a.crispin@mmu.ac.uk, k.crockett@mmu.ac.uk

Abstract. In this paper a greedy randomized heuristic is used to solve a service technician routing and scheduling problem with time windows. Time window constraints occur in many sectors such as telecommunications, maintenance, call centres, warehouses and healthcare, and is a way of service providers differentiating themselves to maintain customer satisfaction and ultimately retain market share. The greedy randomized heuristic is coupled with a simulated annealing with restart metaheuristic and tested on 36 problem instances. The greedy randomized heuristic is compared against a parallel adaptive large neighbourhood search heuristic, presenting new best known results in 18% of the datasets.

Keywords: Technician routing and scheduling problem with time windows · Greedy randomized heuristic · Simulated annealing with restart

1 Introduction

The service technician routing and scheduling problem with time windows (STRSPTW) is an NP-hard combinatorial optimisation problem, meaning that as the problem size increases, exact methods become prohibitive, and, therefore, approximate techniques must be used in order to find feasible and high quality solutions within reasonable computational times. This research is based on a combinatorial optimisation problem applicable to many industries such as the maintenance sector (Fırat and Hurkens, 2012 and Pillac et al., 2013), call centres (Van den Bergh et al., 2013) and home healthcare (Hiermann et al., 2015 and Paraskevopoulos et al., 2016).

In the context of STRSPTW, it is not solely the cost of the employees, but also the maintenance and repair of the fleet of vehicles that must be considered. Efficient scheduling and routing of employees and vehicles can reduce the cost of a workforce, ensure a balanced workload, and has the potential to reduce the environmental impacts caused by the vehicles used.

The occurrence of time windows is becoming increasingly popular with service maintenance providers, and directly affects the scheduling and routing of employees. From a customer's perspective, knowing that a technician/skilled worker will be arriving between time a_i and b_i can improve the customer experience. It may allow the customer waiting for a service to take less time off work, and even choose a preferred time slot, providing not only convenience but satisfaction.

In this paper, the first sequential heuristic to solve the STRSPTW data instances, that were first proposed by Kovacs et al., (2012), is described. The problem instances were adapted from the vehicle routing instances proposed by Solomon, (1987), by combining the datasets with skill domain information taken from the ROADEF 2007 challenge (Society, 2016). The STRSPTW datasets each have 100 customers, a varying crew size, and different proportions of jobs with time windows. Kovacs et al., (2012) used a parallel adaptive large neighbourhood search (pALNS) algorithm. This research proposes a sequential greedy randomized heuristic on the problem instances to provide a comparative performance analysis.

The rest of this paper is organised as follows: section 2 provides a literature review of research in the field of personnel scheduling, section 3 presents the mathematical formulation of the STRSPTW, section 4 describes the greedy randomized heuristic approach, section 5 presents the results of the computational experiments performed, section 6 discusses the results obtained and section 7 concludes on the research undertaken and suggests further areas for investigation.

2 Literature

Research in the area of personnel scheduling has included a diverse range of constraints such as routing, teaming, priority levels, precedence, time windows, multi-period (multiple days), tools and spare parts, and experienced based service times. However, the main characteristic featured in all of these problems is skill complexity. All of these problems require jobs to be completed by a technician/team who collectively posses the necessary expertise to perform the required job.

A study by Pillac et al., (2012) concentrated on the dynamic technician routing and scheduling problem (DTRSP), in which new job requests appear, an aspect of a real world situation faced by industry. In addition, a static TRSP was also studied by Pillac et al., (2013, who extended instances from vehicle routing problems proposed by Solomon [7] by combining them with randomly generated skill requirements and tools and spare parts information. This TRSP used a crew of up to 25 technicians and scheduled up to 100 jobs and required the scheduling of crew over a single day. This research included the complexity of tools and spare parts constraints, an important aspect in the service maintenance sector.

An exact approach was studied by Tricoire et al., (2013) who compared the performance of exact and hybrid solution approaches, concluding the trade off between computational time and solution quality. Chen et al., (2016) studied a version of the TRSP where the technicians became more experienced throughout the scheduling horizon resulting in a reduction of service times, an aspect of the real world scenario previously unstudied.

Other exact approaches in the literature include papers by Mathlouthi et al., (2016) and Zamorano and Stolletz, (2016 who used mixed integer programming (MIP) and branch and price solution approaches respectively. Mathlouthi et al., (2016) used artificial datasets that contained up to 25 jobs and used CPLEX to solve the mixed integer programming model. The problem included the complexities such as skill requirements, priority levels, time windows, breaks and overtime. This paper also illustrated

how the computational time needed to solve the problems rapidly increases with problem size and complexity. Zamorano and Stolletz, (2016 used both artificial and real world datasets containing up to 29 jobs, again emphasizing the difficulties faced with the scalability of Mixed Integer Programming (MIP) solution approaches.

Throughout the literature it is clear that research is needed into approximate approaches in order to tackle medium and large scale problems as exact methods are prohibitive. This research focuses on designing and implementing approximate approaches that can deal with relevant real world constraints such as time windows. As evidenced, time windows are an important consideration for service providers who seek to maintain customer satisfaction and maintain repeat business through providing a reliable and customer focused service.

3 Problem formulation

The MIP formaulation of the STRSPTW datasets, as introduced by Kovacs et al. [6] is presented below. The problem can be defined mathematically as a complete directed graph $G = \{V, A\}$, where V is the set of all vertices i.e the set of jobs, and A a set of arcs between the vertices. The set of jobs that is allocated can be defined as V', and jobs that belong to V but not V' is the set of jobs that is outsourced.

There is a set of technicians $T = \{1, \ldots, t\}$ The starting depot is denoted as 0 and the ending depot as N. Each technician has intrinsic skills $s \in S$ and varying levels $l \in L$ within each area of expertise. The fleet of technicians is heterogeneous, each being unique with different skills and levels within each skill area. The technician's skills can be represented by an $L \times S$ matrix where $[p^t_{l,s}]$ denotes the level of expertise the technician has in skill area s to level l. Skill levels are hierarchical so if $p^t_{l,s} = 1$ then $p^t_{l',s} = 1$ for $l' < l$.

Each job belonging to the set V has a service time denoted by d_i, a skill requirement matrix, of size $L \times S$, denoted as $q_{l,s}$ and a time window in which service of the job must begin $[a_i, b_i]$. The aim of the problem is to construct the least costly schedule, by minimizing the total sum of the outsourcing and routing costs. The following variables are used;

$B^t_i =$ the beginning time of service of job i or the depot by technician t
$E^t_i =$ the end time of service of job i or the depot by technician t
$x^t_{i,j}$ that equals one only if technician t travels from job i to job j
z_i equals one only if job i has been outsourced
y^t_i equals one if job i is assigned to technician t

The problem can now be represented as;

$$\min \sum_{t \in T} \sum_{i,j \in A} c_{i,j} \cdot x^t_{i,j} + \sum_{i \in V'} o_i \cdot z_i \tag{1}$$

Equation (1) shows the objective function of the problem, which is to minimise the total sum of the routing costs ($c_{i,j}$ is the distance between customers i and j) and the outsourcing costs (where o_i is the cost of outsourcing job i). Subject to;

$$\sum_{t \in T} y_i^t + z_i = 1 \quad \forall i \in V \tag{2}$$

Equation (2) guarantees that each job is either outsourced or it is allocated to a technician.

$$\sum_{j \in V' \cup \{N\}} x_{0,j}^t = 1 \quad \forall t \in T \tag{3}$$

$$\sum_{i \in V' \cup \{N\}} x_{i,N}^t = 1 \quad \forall t \in T \tag{4}$$

Equations (3 and 4) ensure each technician departs from the central depot at the beginning of the working day and returns to the depot at the end of the working day.

$$\sum_{j \in V' \cup \{0\}} x_{j,i}^t = y_i^t \quad \forall i \in V', t \in T \tag{5}$$

$$\sum_{j \in V' \cup \{0\}} x_{j,i}^t - \sum_{j \in V'} x_{i,j}^t = 0 \quad \forall i \in V', t \in T \tag{6}$$

Equations (5 and 6) confirm that if a technician is assigned to a job i, then the technician must travel to the job from another location, and leave the job to travel to another location.

$$B_j^t \geq (E_i^t + c_{i,j}) * x_{i,j,}^t \quad \forall i \in V', t \in T \tag{7}$$

Equation (7) states that if two jobs i and j happen sequentially, then the start time of j must be equal to or greater than the end time of i plus that travel time between i and j, to ensure continuity.

$$B_0^t = 0 \quad \forall t \in T \tag{8}$$

Equation (8) sets the beginning time of each technician's route.

$$y_i^t \cdot q_{l,s}^i \leq p_{l,s}^t \quad \forall i \in V', \quad l \in L, \quad s \in S \tag{9}$$

Equation (9) ensures that if a job is allocated to a technician, the technician has the skills necessary to service the job.

$$E_i^t = (B_i^t + d_i) \cdot y_i^t \quad \forall i \in V', t \in T \tag{10}$$

Equation (10) states that the end service time of a job must be equal to the beginning service time plus the service time of the job.

$$a_i \leq B_i^t \leq b_i \quad \forall i \in V' \cup \{N\}, t \in T \tag{11}$$

Equation (11) guarantees that the beginning of service of a job i is within the time window.

$$x_{i,j,}^t \in \{0,1\} \quad \forall (i,j) \in A, t \in T \tag{12}$$

$$z_i \in \{0,1\} \quad \forall i \in V' \tag{13}$$

$$y_i^t \in \{0,1\} \quad \forall i \in V', t \in T \tag{14}$$

$$B_i^t \geq 0 \quad \forall i \in V' \tag{15}$$

$$E_i^t \geq 0 \quad \forall i \in V' \tag{16}$$

4 Heuristic approach

The approach presented in this paper comprises of two parts, generating an initial feasible solution, and secondly, iteratively trying to improve the current solution through the use of local operators and evaluating using a simulated annealing with restart metaheuristic.

4.1 Greedy randomized construction heuristic

The greedy randomized heuristic behaves in a flexible manner by changing the sorting criteria that decide which job is next to be allocated. There are five insertion methods; earliest late window, minimum window size, complex jobs, depot distance and random. Each of these methods is described in the following subsections. The pseudo code for the greedy randomized construction heuristic is displayed in Figure 1.

– **Earliest late window (ELW)** This insertion method orders the set of unallocated jobs into a list. Each job has a time window $[a_i, b_i]$, and the sorting method orders the jobs into an increasing order of b_i, i.e the end of the time window, the latest time the job can be started. This method tries to ensure that all jobs are allocated before their time window has passed, as they will then be outsourced in order to stay within the feasible solution space, which incurs a cost.

$$earlylate_i = b_i \tag{17}$$

– **Minimum window size (MWS)** The minimum window size method orders the set of unallocated jobs into a list, sorting them by the size of their time window. This method aims to ensure that jobs that have a small opportunity to be started, i.e the difference between b_i and a_i is small, have a higher chance of being allocated in favour of jobs with a larger difference between b_i and a_i.

$$minwindow_i = b_i - a_i \tag{18}$$

– **Complex jobs (CJ)** The set of unallocated jobs is ordered by the complexity of the jobs that are currently unallocated. The difficulty of a job is calculated as the sum of the total skill requirements across each domain and skill level, as in Cordeau et al. [14]. This method aims to allocate jobs which require lots of skill earlier, and jobs that are less difficult to schedule are scheduled later.

$$complex_i = \sum_{l \in L} \sum_{s \in S} q_{l,s}^i \tag{19}$$

– **Depot distance (DD)** The depot distance method orders the set of jobs in ascending order of distance away from the depot. The distance between a job i located at x_i, y_i, and the depot located at x_0, y_0 is calculated using the Euclidean distance as shown in Equation (20).

$$depotdistance_i = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2} \tag{20}$$

– **Random (R)** The random sorting method orders the unallocated jobs by shuffling the array that contains the jobs. In this work, the level of randomness, r, is set to 0.08.

Greedy randomized heuristic pseudo code Figure 1 shows the greedy randomized construction heuristic. This algorithm takes the following variables, a set of jobs V, a set of teams τ, a schedule S, an outsourcing list O, the following sorting methods ELW, MWS, CJ, DD, and R. The job selected for allocation is i, and τ_i the team selected to be allocated job i. allocated job i.

Variables: V: set of jobs, τ: the set of teams , S: the schedule, O: outsource list, ELW: earliest late window, MWS: minimum window size, CJ: complex jobs, DD: depot distance, R: Randomly, i:job selected for allocation, τ_i: team selected to serve job i

```
1: initialise  S
2: while jobs can be allcoated  do
3:    r ← random(0,1)
4:    technique ← Choosesortingmethod(r,ELW,MWS,CJ,DD,R)
5:    V ← method(V)
6:    i ← select job(V)
7:    τ_i ← selectteam(i,S)
8:    assign(S,τ_i,i)
9:    remove(V,i)
10: end while
11: O ← addOutsourced(V)
12: return S
```

Fig. 1: Greedy randomized construction heuristic

First, an empty schedule S is initialised. While jobs can be allocated to the schedule S, a random number r is generated on the interval $\{0,1\}$. Dependent on the value of r, a sorting method is chosen; ELW earliest late window, MWS minimum window size, CJ complex jobs, DD depot distance or R randomly. On line 5, the set of remaining unallocated jobs V is sorted by the chosen sorting method, then a job i is selected belonging to V. Job i is then assigned to a team (if one is available in terms of time windows and skill requirements) and removed from the set of unallocated jobs. The while loop is iterated through until no more job allocations can be made to the teams. On line 11, any remaining unallocated jobs are added to the outsource list O. Lastly, on line 12, the initial solution S is output.

The cost of outsourcing a job is shown in equation (21). Note, it is always less costly to schedule a job, if there are technicians available rather than to outsource it.

$$o_i = 200 + \sum_{l \in L} \sum_{s \in S} q^i_{l,s} \tag{21}$$

4.2 Simulated annealing with restart

In this work a simulated annealing metaheuristic with a restart mechanism has been implemented. Simulated annealing was chosen due to its success in other types of combinatorial optimisation problems(Kundu et al., (2008) and Cordeau et al., 2010. The implementation of this metaheuristic is shown in Figure 2.

Variables: S: current solution, S': neighbouring solution, S_{Best}: the best solution, O: the set of local operators, T : initial temperature, δT: the cooling rate, $StepSize$: maximum steps before beginning from best solution, $count$: counter for iterations,

---

$S_{Best} \leftarrow S$
$count \leftarrow 0$
while termination criteria not met **do**
  $randomly \quad choose \quad o \in O$
  $S' \leftarrow o(S)$
  if $S' \leq S$ **then**
    $S \leftarrow S'$
    if $S \leq S_{Best}$ **then**
      $S_{Best} \leftarrow S$
      $count \leftarrow 0$
    end if
  else
    $r \leftarrow random(0,1)$
    $p \leftarrow exp^(S' - S)/T$
    if $p \geq r$ **then**
      $S \leftarrow S'$
    end if
  end if
  $T \leftarrow T \cdot \delta T$
  $count \leftarrow count + 1$
  if $count = StepSize$ **then**
    $S \leftarrow S_{Best}$
    $count \leftarrow 0$
  end if
end while
return S_{Best}

Fig. 2: Simulated annealing with restart metaheuristic

The variables associated are: S the initial solution generated by the greedy randomized construction heuristic, S' the neighbouring solution generated by applying a local operator to S, S_{Best} the best solution found, O the set of local operators which perturb the solution S, T the initial temperature, δT the decrement rate, $StepSize$ the maximum number of steps before restarting from the best solution, and lastly, *count* which counts the number of iterations.

The initial solution S, generated by the greedy randomized heuristic, is saved as the best solution S_{Best} on line 1, and *count* is set to 0. Whilst the termination criterion is not met, i.e. there is computational time remaining, a local operator is selected on line 4. This local operator o is applied to the solution S on line 5 generating a neighbouring solution S'. On line 6 this solution S' is evaluated. If it has a lower objective function than S, then it replaces S. Next, on line 8 the solution S is evaluated against the best solution S_{Best} and if better, the best solution is updated and the *count* is set to zero. However, if solution S' is not better than the current solution S then it is evaluated using the simulated annealing criterion and compared to a random number r generated on the interval (0,1). If the probability p of accepting this solution is greater than r then solution S is updated. After every iteration, the simulated annealing temperature is reduced and the *count* is incremented by one. Once the *count* has reached its maximum value, $StepSize$, solution S is set to S_{Best} on line 22, and the *count* is set back to 0. Once the termination criterion has been met, the best solution S_{Best} is output on line 26.

5 Computational experiments

The greedy randomized heuristic was programmed in Java and tested on an HP Z210 Workstation, with an i7-2600 CPU with 3.4 GHZ with 12GB of RAM. Each run lasted 80 seconds as in Kovacs et al., (2012 for comparison purposes. The greedy randomized heuristic was run 5 times per data instance, and the best, average and worst results obtained are shown in Table 1.

Column 1 shows the dataset, columns 2-4 show the best, average and maximum objective value achieved by Kovacs et al., (2012 with the pALNS, and columns 4-7 displays the best, average and maximum objective values found by the greedy randomized heuristic. The highlighted rows indicate where the greedy randomized heuristic has found a lower objective value than the pALNS.

6 Discussion

6.1 Performance of greedy randomized heuristic on *NoTeam* instances

Table 1 displays the results achieved for the *NoTeam* problem instances. In these datasets, the sequential greedy randomized heuristic is able to find a lower minimum objective value than the pALNS in 8 out of 36 datasets and is able to find the same minimum objective value in two datasets, $C201\_6 \times 6\_NoTeam$ and $C101\_7 \times 4\_NoTeam$.

The results illustrate that the greedy randomized heuristic finds a smaller gap from BKS on the *01* instances compared to the *03* instances. The difference between these

Table 1: *NoTeam*

Dataset	pALNS			GREEDY		
	min	avg	max	min	avg	max
*C*101_5 × 4_*NoTeam*	1098.71	1111.08	1128.02	1096.85	1135.03	1180.95
*C*103_5 × 4_*NoTeam*	1018.61	1037.33	1049.41	1075.36	1119.66	1195.76
*C*201_5 × 4_*NoTeam*	1158.97	1180.93	1228.99	1157.65	1163.1	1183.31
*C*203_5 × 4_*NoTeam*	1046.93	1049.3	1052.83	1228.23	1297.39	1337.33
*R*101_5 × 4_*NoTeam*	1678.68	1685.85	1697.2	1672.55	1682.17	1692.81
*R*103_5 × 4_*NoTeam*	1238.67	1249.91	1282.28	1288.48	1312.95	1339.13
*R*201_5 × 4_*NoTeam*	1440.3	1448.93	1462.62	1526.43	1563.53	1599.98
*R*203_5 × 4_*NoTeam*	1098	1106.12	1123.08	1281.83	1334.01	1378.83
*RC*101_5 × 4_*NoTeam*	1708.51	1716.07	1729.75	1676.57	1721.95	1760.88
*RC*103_5 × 4_*NoTeam*	1337.99	1354.11	1388.13	1454.46	1482.46	1507.06
*RC*201_5 × 4_*NoTeam*	1601.89	1607.25	1610.75	1650.66	1698.03	1727.49
*RC*203_5 × 4_*NoTeam*	1161.53	1166.5	1178.64	1373.44	1430.32	1467.19
*C*101_6*x*6_*NoTeam*	989.21	1004.82	1029.72	973.05	1002.15	1029.72
*C*103_6 × 6_*NoTeam*	893.94	897.86	907.62	1075.26	1181.12	1239.93
*C*201_6 × 6_*NoTeam*	821.55	821.55	821.55	821.55	847.22	868.72
*C*203_6 × 6_*NoTeam*	689.6	703.1	750.12	831.51	908.91	970.66
*R*101_6 × 6_*NoTeam*	1658.27	1667.43	1672.57	1662.69	1666.02	1675.24
*R*103_6 × 6_*NoTeam*	1223.63	1231.49	1243.49	1243.7	1264.54	1286.5
*R*201_6 × 6_*NoTeam*	1261.94	1270.26	1279.81	1335.66	1375.56	1417.77
*R*203_6 × 6_*NoTeam*	932.35	951.84	964.54	1104.75	1153.24	1200.86
*RC*101_6 × 6_*NoTeam*	1679.13	1683.96	1690.06	1672.85	1686.62	1693.34
*RC*103_6 × 6_*NoTeam*	1281.55	1310.95	1331.46	1354.14	1381.79	1400.85
*RC*201_6 × 6_*NoTeam*	1395.4	1403.95	1411.48	1494.14	1547.03	1613.75
*RC*203_6 × 6_*NoTeam*	1001.04	1016.71	1030.15	1176.41	1236.67	1291.41
*C*101_7 × 4_*NoTeam*	1357.05	1398.95	1462.16	1357.05	1416.19	1553.71
*C*103_7 × 4_*NoTeam*	1215.7	1239.22	1264.17	1263.67	1295.83	1335.95
*C*201_7 × 4_*NoTeam*	1256.56	1282.18	1302.56	1256.3	1264.26	1302.56
*C*203_7 × 4_*NoTeam*	1150.85	1151.27	1152.94	1288.96	1354.81	1474.64
*R*101_7 × 4_*NoTeam*	1776.46	1793.95	1813.53	1771.56	1791	1807.89
*R*103_7 × 4_*NoTeam*	1346.8	1375.09	1399.95	1402.04	1423.96	1456.49
*R*201_7 × 4_*NoTeam*	1398.14	1410.9	1427.95	1427.56	1458.63	1474.29
*R*203_7 × 4_*NoTeam*	1164.9	1166.94	1169.27	1285.35	1334.17	1407.76
*RC*101_7 × 4_*NoTeam*	1821.9	1844.37	1859.17	1832.75	1903.58	1980.33
*RC*103_7 × 4_*NoTeam*	1435.63	1455.33	1477.84	1547.33	1610.13	1679.64
*RC*201_7 × 4_*NoTeam*	1697.82	1701.25	1705.48	1771.25	1793.66	1811.06
*RC*203_7 × 4_*NoTeam*	1239.45	1241.65	1249.72	1422.35	1459.22	1527.29

datasets is the proportion of time windows, the *01* instances are more constrained (contain 100% time windows) compared to the *03* instances (contain 50% time windows) and therefore, there are fewer feasible options of when to allocate a job. In the 5×4, 6×6 and 7×4 datasets, the gap from minimum objective results in the *01* instances

is 1.08%, 1.98% and 1.12%. This increases to 11.77%, 14.03% and 8.82% in the *03* instances.

Another trend within the results occurs in the *203* datasets. These datasets achieve the highest gap from BKS overall, regardless of the distribution of job locations i.e *C* clustered, *R* randomly, *RC* randomly clustered. This pattern occurs across each set of domains and levels, 5×4, 6×6 and 7×4.

Furthermore, another pattern across the distribution of customers' locations is evident in the 5×4 and 7×4 datasets. Throughout the distributions, C, R and RC, the gap from BKS increases. In 5×4 the gap is 5.75%, 6.59% and 7.03% and in 7×4 the gap is 3.98%, 4.07% and 6.86%, respective of distributions R,C and RC.

The results here demonstrate that as the problem instances become more complex, in terms of the number of skill domain areas, or the proportion of time windows, the greedy randomized heuristic is unable to match the performance of the pALNS.

7 Conclusion

This research has presented an approach to solving the STRSPTW, and 8 new best known results for these datasets have been found. It is evident that the quality of solution produced by the greedy randomized heuristic is heavily dependent on the proportion of time windows within the datasets. Future work will investigate enhancements that could be made to this heuristic in order to improve performance on the '03' datasets.

The parallels the STRSPTW has with other domains should not be underestimated, there are many constraints and complexities that are shared with other problems such as the home healthcare problem.

Future research will focus on improving the performance of the greedy randomized heuristic on these instances, and also investigating other common complexities in related service maintenance combinatorial optimisation problems such as tools and spare parts.

Bibliography

[1] Murat Fırat and CAJ Hurkens. An improved mip-based approach for a multi-skill workforce scheduling problem. *Journal of Scheduling*, 15(3):363–380, 2012.

[2] Victor Pillac, Christelle Gueret, and Andrés L Medaglia. A parallel matheuristic for the technician routing and scheduling problem. *Optimization Letters*, 7(7): 1525–1535, 2013.

[3] Jorne Van den Bergh, Jeroen Beliën, Philippe De Bruecker, Erik Demeulemeester, and Liesje De Boeck. Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3):367–385, 2013.

[4] Gerhard Hiermann, Matthias Prandtstetter, Andrea Rendl, Jakob Puchinger, and Günther R Raidl. Metaheuristics for solving a multimodal home-healthcare scheduling problem. *Central European Journal of Operations Research*, 23(1): 89–113, 2015.

[5] Dimitris C Paraskevopoulos, Gilbert Laporte, Panagiotis P Repoussis, and Christos D Tarantilis. Resource constrained routing and scheduling: Review and research prospects. 2016.

[6] Attila A Kovacs, Sophie N Parragh, Karl F Doerner, and Richard F Hartl. Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of scheduling*, 15(5):579–600, 2012.

[7] Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

[8] French Operational Research Society. What is the roadef 2007 challenge, 2016. URL http://challenge.roadef.org/2007/en/.

[9] Victor Pillac, Christelle Guéret, and Andrés Medaglia. On the dynamic technician routing and scheduling problem. 2012.

[10] Fabien Tricoire, Nathalie Bostel, Pierre Dejax, and Pierre Guez. Exact and hybrid methods for the multiperiod field service routing problem. *Central European Journal of Operations Research*, 21(2):359–377, 2013.

[11] Xi Chen, Barrett W Thomas, and Mike Hewitt. The technician routing problem with experience-based service times. *Omega*, 61:49–61, 2016.

[12] Ines Mathlouthi, Michel Gendreau, and Jean-Yves Potvin. Mixed integer programming for a multi-attribute technician routing and scheduling problem. 2016.

[13] Emilio Zamorano and Raik Stolletz. Branch-and-price approaches for the multiperiod technician routing and scheduling problem. *European Journal of Operational Research*, 2016.

[14] Jean-François Cordeau, Gilbert Laporte, Federico Pasin, and Stefan Ropke. Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, 13(4):393–409, 2010.

[15] S Kundu, M Mahato, B Mahanty, and S Acharyya. Comparative performance of simulated annealing and genetic algorithm in solving nurse scheduling problem. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, pages 96–100, 2008.